22.4.6 Klassen Table, Table.Fields und Table.Indexes (gb.db)

- Die Klasse Table repräsentiert u.a. die Definition einer Datenbank-Tabelle.
- Mit den Methoden der Klasse Table.Fields können Sie Felder in einer DB-Tabelle definieren.
- Mit den Methoden der Klasse Table.Indexes können Sie einer DB-Tabelle einen Index hinzufügen.

22.4.6.1 Eigenschaften Table

Die Klasse *Table* verfügt über diese Eigenschaften:

Eigenschaft	Datentyp	Beschreibung
Connection	Connection	Gibt das übergeordnete Connection-Objekt einer Tabelle zurück.
Name	String	Gibt den Namen einer DB-Tabelle zurück.
PrimaryKey	String[ ]	Gibt den Primärschlüssel einer DB-Tabelle als String-Array zurück oder legt den Primärschlüssel fest. Dieser Primärschlüssel ist ein String-Array mit den Namen der einzelnen Primärschlüsselfelder.
System	Boolean	Liefert True, wenn es sich bei der Tabelle um eine Systemtabelle handelt, d.h. um eine vom Datenbankserver (intern) verwendete Tabelle.
Type	String	Gibt den Typ einer DB-Tabelle zurück oder setzt diesen Typ. Diese Eigenschaft wird nur für MySQL-DB-Tabellen benutzt. Typen sind u.a. MyISAM, InnoDB oder BDB.
Indexes	Table.Indexes	Gibt eine Übersicht (Datentyp .Table.Indexes) der Indizes einer DB-Tabelle zurück. Der Primärschlüssel ist mit Abstand der wichtigste Index in einer Tabelle.
Fields	Table.Fields	Gibt eine Übersicht (Datentyp Table.Fields) zu den Feldern einer DB-Tabelle zurück.

Tabelle 22.4.6.1.1 : Eigenschaften der Klasse Table

Hinweis

Für Primärschlüssel (primary key) gilt wegen der Eindeutigkeit, dass sein Wert nie undefiniert sein darf (auch NULL ist nicht zulässig) und sein Wert darf nur einmal vorkommen. Wenn ein Primärschlüssel als zusätzliches Attribut in eine andere Tabelle übernommen wird, so nennt man ihn *Fremdschlüssel*. Er kann je Tabelle einmal verwendet werden und darf dabei über einer Spalte als auch über mehrere Spalten definiert sein. Der Primärschlüssel ist auch der wichtigste Index in einer Tabelle.

Beispiel

```
Dim iCount As Integer, vElement As Variant, hDBTable As Table

hDBTable = DBCS.DBConnection.Tables["contacts"]

Print "DBTableName: "; hDBTable.Name
Print "DBName: "; hDBTable.Connection.Name
Print "DBPrimaryKey: "; IIf(hDBTable.PrimaryKey.Count = 0, "Kein primärer Schlüssel definiert.",
                            "Es existiert ein primärer Schlüssel.")
For Each vElement In hDBTable.PrimaryKey
  Print "Primäres Schlüsselfeld "; iCount; ": "; vElement
  Inc iCount
Next
Print "DBSystem: "; IIf(hDBTable.System, "Tabelle ist DBSystem-Tabelle",
                         "Tabelle ist keine DBSystem-Tabelle.")
Print "DBTabellentyp: "; IIf(hDBTable.Type, hDBTable.Type,
                              "Tabelle ist keine MySQL-Tabelle - kein Typ definiert.")
```

Das sind die Ausgaben in der Konsole der IDE:

```
DBTableName: contacts
DBName: contacts.sqlite
DBPrimaryKey: Es existiert ein primärer Schlüssel.
Primäres Schlüsselfeld 1: id
DBSystem: Tabelle ist keine DBSystem-Tabelle.
DBTabellentyp: Tabelle ist keine MySQL-Tabelle - kein Typ definiert.
```

22.4.6.2 Methode Table

Die Klasse Table verfügt nur über eine Methode: Sub Update( ). Sie legt eine DB-Tabelle in der aktuellen Datenbank an, deren Felder, Primärschlüssel und Tabellen-Type – nur bei MySQL – vorher definiert wurden. Ein Tabellen-Index auf ein oder mehrere Felder wird separat erzeugt.

```
Dim hDBTable As Table
  Dim hIndex As Index
  Dim sTableIndexName As String = "index_lastname_members"
  Dim iCount As Integer = 1


  If Not DBCS.DBConnection.Tables.Exist("members") Then

'--- DB-Tabellen-Objekt erzeugen - Tabellenname: `members`
     hDBTable = DBCS.DBConnection.Tables.Add("members")

'--- DB-Felder definieren
     With hDBTable.Fields
       .Add("m_id", db.Serial) ' gb.Serial => INTEGER PRIMARY KEY AUTOINCREMENT
       .Add("lastname", db.String)
       .Add("date", db.Date)
       .Add("description", db.String)
     End With

'--- Primärschlüssel definieren » Feldname: `m_id`
     hDBTable.PrimaryKey = ["m_id"]

'--- DB-Tabelle `members` in der Datenbank `contacts.sqlite` erzeugen
     hDBTable.Update()

'--- DBIndex definieren » Indexname: `index_lastname_members`, Feldname: `lastname`
     If Not hDBTable.Indexes.Exist("index_lastname_members") Then
        hDBTable.Indexes.Add("index_lastname_members", ["lastname"])
     Endif

  Else
'--- Die existierende DBTabelle `members` wird aktuelle DBTabelle
     hDBTable = DBCS.DBConnection.Tables["members"]

  Endif
```

22.4.6.3 Eigenschaften Table.Fields

Die Klasse besitzt nur eine (lesbare) Eigenschaft: Table.Fields.Count vom Datentyp Integer. Sie gibt die Anzahl der Felder in einer (existierenden) DB-Tabelle zurück.

22.4.6.4 Methoden Table.Fields

Mit den Methoden der Klasse Table.Fields können Sie neue Felder in einer DB-Tabelle definieren. Diese Klasse verhält sich wie ein schreibgeschütztes Array und ist mit dem Schlüsselwort FOR EACH enumerierbar.

Methode	Rückgabetyp	Beschreibung
Table.Fields.Add (Name As String, Type As Integer [, Length As Integer, Default As Variant])	-	Fügt ein neues Feld in eine DB-Tabelle ein. `Name` ist der Name des Feldes. `Type` ist sein Datentyp. Die möglichen Feld-Datentypen sind: db.Blob (-2), db.Boolean (1), db.Date (8), db.Float (7), db.Integer (4), db.Long (5), db.Serial (-1) und db.String (9). Hinweis zu `db.Serial`: Diese Konstante steht für einen Serien-Feld-Datentyp. Ein serielles Feld ist ein Feld, dessen Wert einzigartig ist und automatisch bei der Erzeugung von jedem neuen Datensatz erhöht wird (Auto-Increment). `Length` ist die maximale Länge für Textfelder. `Default` ist ein weiterer optionaler Parameter, der dem Default-Wert des zu Grunde liegenden Feld-Typs entspricht. Wenn er nicht angegeben wird, so ist der Standardwert 0.
Table.Fields.Exist (Key As String)	Boolean	Liefert True, wenn das angegebene Feld in der DB-Tabelle existiert.
Table.Fields.Refresh ( )	-	Aktualisiert die Feld-Übersicht (Datentyp Table.Fields) einer DB-Tabelle durch Löschen des internen Caches.

Tabelle 22.4.6.4.1 : Methoden der Klasse Table.Fields

22.4.6.5 Eigenschaften Table.Indexes

Die Klasse besitzt nur eine Eigenschaft: *Property Read Table.Indexes.Count As Integer*. Sie gibt die Anzahl der indizierten Felder in einer DB-Tabelle zurück.

22.4.6.6 Methoden Table.Indexes (gb.db)

Feldern einer DB-Tabelle können Sie mit den Methoden der Klasse Table.Indexes ein Index hinzufügen. Die Klasse *Table.Indexes* verfügt über vier Methoden.

Methode	Rückgabetyp	Beschreibung
Sub Table.Indexes.Add (Name As String, Fields As String[] [, Unique As Boolean])	-	Fügt einen neuen Index in eine bestehende Tabelle ein. `Name` ist der Name des Index. `Fields` ist ein String-Array. Der optionale Parameter `Unique` zeigt an, ob dieser Index eindeutig sein muss oder nicht. Standardmäßig ist der Index nicht eindeutig.
Table.Indexes.Exist (Key As String)	Boolean	Gibt True zurück, wenn der angegebene Index in der DB-Tabelle existiert, sonst False.
Table.Indexes.Refresh ( )	-	Aktualisiert die Index-Übersicht (Datentyp Table.Indexes) einer DB-Tabelle durch Löschen des internen Caches.
Table.Indexes.Remove (Name As String)	-	Löscht den angegebenen Index in einer DB-Tabelle.

Tabelle 22.4.6.6.1 : Methoden der Klasse Table.Indexes